Ex.9 Clustering of Data using K-means Clustering Technique

October 26, 2023

[ ]:
```
Bewin Felix R A
URK21CS1128
```

[ ]:
```
AIM:
    To apply K-means clustering technique to the given dataset.
```

[ ]:
```
DESCRIPTION:

K-means clustering is a popular unsupervised machine learning
technique used to partition a dataset into K distinct, non-overlapping
clusters based on the similarity of data points. It aims to group similar
data points together and find cluster centroids that minimize the
within-cluster variance.
from sklearn.cluster import KMeans
# Choose the number of clusters (K)
k = 3
# Initialize the K-means model
kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10,
random_state=0)
# Fit the model to your data
kmeans.fit(X)
# Get cluster labels for each data point
labels = kmeans.labels_
# Get cluster centroids
centroids = kmeans.cluster_centers_
Elbow Method for Optimal K:
Description: The Elbow Method helps in determining the optimal number of
clusters (K) for K-means. It involves fitting K-means with different K
values and plotting the Within-Cluster Sum of Squares (WCSS) against K.
wcss = []
for k in range(1, 11):
kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10,
random_state=0)
kmeans.fit(X)
wcss.append(kmeans.inertia_)
# Plot WCSS vs. K
# ...
Visualization of Clusters:
```

```
Description: Visualizing clusters helps in understanding the clustering results.
 You can create scatter plots with data points colored by their cluster
assignments.
plt.scatter(X['Age'], X['Annual_Income'], c=kmeans.labels_, cmap='rainbow')
plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c='black', marker='X',
label='Centroids')
plt.title('K-means Clustering')
plt.xlabel('Age')
plt.ylabel('Annual Income')
plt.legend()
plt.show()
6. Performance Metrics for Different K-values
Description: You calculate performance metrics (silhouette_score and
davies_bouldin_score) for different values of K to evaluate the quality of
clustering.
k_values = [optimal_k, optimal_k + 1, optimal_k + 2, optimal_k + 3]
silhouette_scores = []
davies_bouldin_scores = []
for k in k_values:
kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10,
                random_state=0)
kmeans.fit(X)
silhouette = silhouette_score(X, kmeans.labels_)
davies_bouldin = davies_bouldin_score(X, kmeans.labels_)
silhouette_scores.append(silhouette)
davies_bouldin_scores.append(davies_bouldin)
```

[ ]:
```
2. Develop a K-means clustering model for the Mall_Customers dataset using the␣
 ↪scikit-
learn.
```

[10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score
```

[11]:
```
print(1128)
df = pd.read_csv('Mall_Customers.csv')
df.head()
```

```
1128
```

[11]:

	CustomerID	Gender	Age	Annual_Income	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77

```
4           5  Female   31              17                      40
```

[12]:
```
#a. Use the columns: ' Age', 'Annual_Income' as the input variables.
print(1128)
X = df[['Age', 'Annual_Income']]
X.head()
```

```
1128
```

[12]:

	Age	Annual_Income
0	19	15
1	21	15
2	20	16
3	23	16
4	31	17

[13]:
```
# b. Compute the optimal number of cluster 'K' from 1-10 using the Elbow method.
print(1128)
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++',␣
 ↪max_iter=300,n_init=10,random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

```
1128
```

[ ]:
```
#c. Plot the graph between number of cluster K and within-cluster sum of␣
 ↪squares (WCSS)
#value.
print(1128)
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS')
plt.grid()
plt.show()
```

[14]:
```
# d. Perform the K-means clustering with the selected optimal K.
print(1128)
optimal_k = 4

kmeans = KMeans(n_clusters=optimal_k, init='k-means++',␣
 ↪max_iter=300,n_init=10,random_state=0)
kmeans.fit(X)
```

```
1128
```

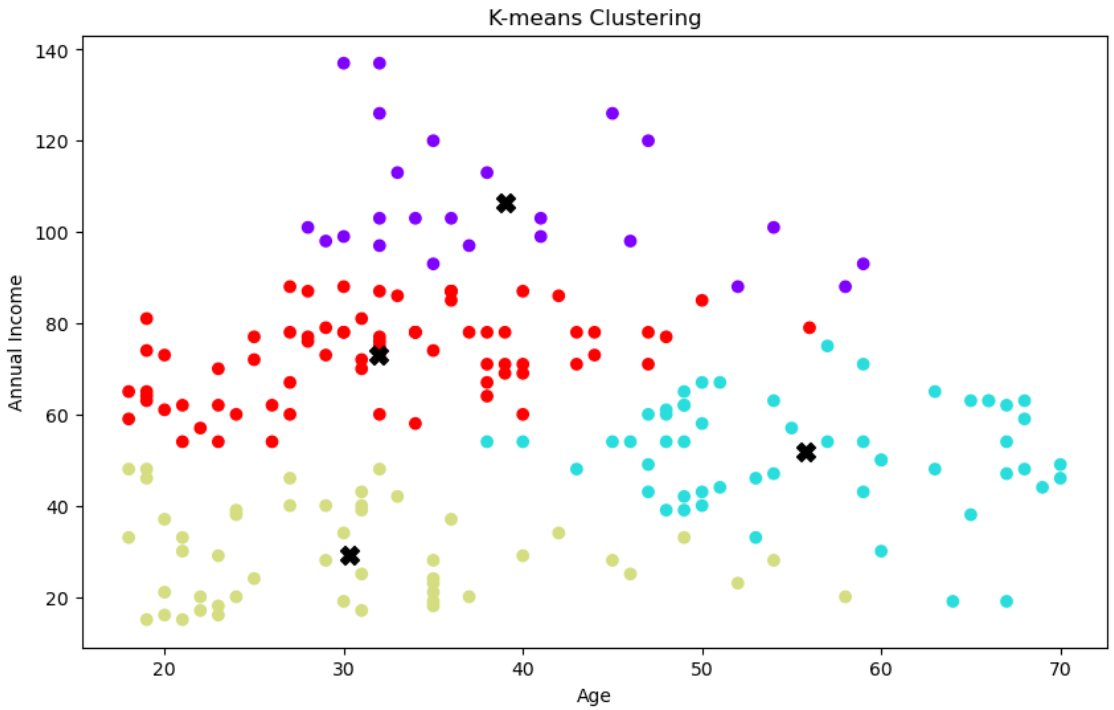
[14]:
```
KMeans(n_clusters=4, n_init=10, random_state=0)
```

[15]:
```
# e. Display the cluster centroids.
print(1128)
centroids = kmeans.cluster_centers_
print("Cluster Centroids:")
print(centroids)
```

```
1128
Cluster Centroids:
[[ 39.         106.5       ]
 [ 55.81481481  51.77777778]
 [ 30.34693878  29.26530612]
 [ 31.95890411  72.95890411]]
```

[16]:
```
# f. Visualize the data representation of K-means clustering.
print(1128)
df['Cluster'] = kmeans.labels_
plt.figure(figsize=(10, 6))
plt.scatter(df['Age'], df['Annual_Income'], c=df['Cluster'], cmap='rainbow')
plt.scatter(centroids[:, 0], centroids[:, 1], c='black', marker='X', s=100)
plt.title('K-means Clustering')
plt.xlabel('Age')
plt.ylabel('Annual Income')
plt.show()
```

```
1128
```



[17]:
```
# g. Change the value of K in K-means with different values and tabulate the performance
#metrics such as silhouette_score and davies_bouldin_score obtained.
print(1128)
k_values_to_evaluate = [optimal_k, optimal_k + 1, optimal_k + 2, optimal_k + 3]
print("K-value\tSilhouette Score\tDavies-Bouldin Score")
for k in k_values_to_evaluate:
    kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300,
                    n_init=10,random_state=0)
    kmeans.fit(X)
    silhouette = silhouette_score(X, kmeans.labels_)
    davies_bouldin = davies_bouldin_score(X, kmeans.labels_)
    print(f"{k}\t{silhouette:.4f}\t\t{davies_bouldin:.4f}")
```

```
1128
K-value Silhouette Score        Davies-Bouldin Score
4       0.4330          0.7696
5       0.4084          0.7552
6       0.3955          0.8186
7       0.3847          0.8472
```

[ ]:
```
RESULT:
    Therefore the K-means clustering technique has been applied to the
    dataset given and the output is displayed successfully.
```